



(86) Date de dépôt PCT/PCT Filing Date: 2003/07/01
(87) Date publication PCT/PCT Publication Date: 2004/01/08
(45) Date de délivrance/Issue Date: 2012/10/30
(85) Entrée phase nationale/National Entry: 2004/12/31
(86) N° demande PCT/PCT Application No.: IB 2003/003259
(87) N° publication PCT/PCT Publication No.: 2004/003741
(30) Priorité/Priority: 2002/07/01 (US10/188,352)

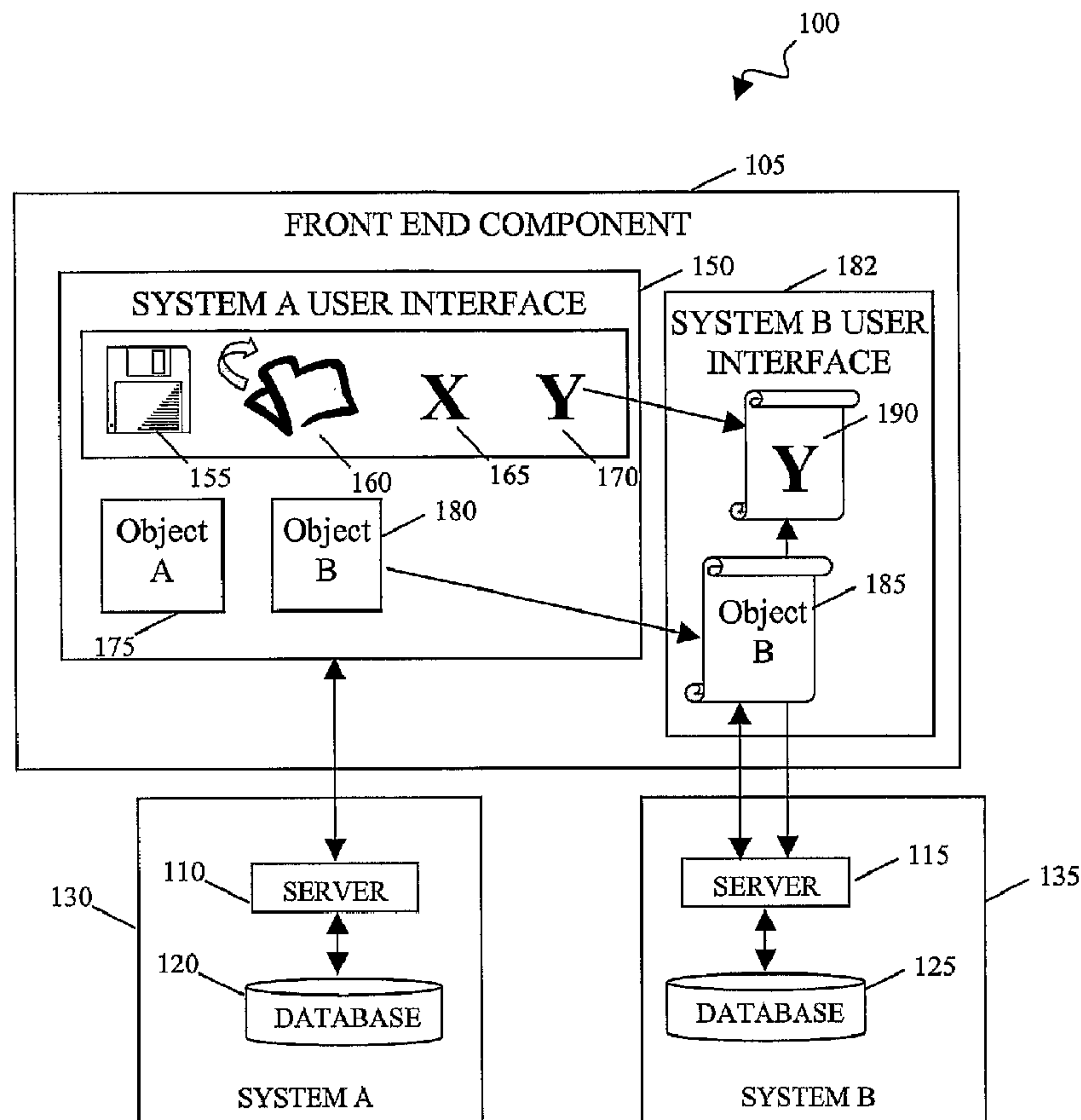
(51) Cl.Int./Int.Cl. *G06F 9/44* (2006.01),
H04L 29/06 (2006.01), *H04L 29/08* (2006.01)

(72) Inventeurs/Inventors:
HAUSER, ANDREAS, DE;
BIHARI, PRANAV, DE;
SCHMITT, BERND, DE;
SZERVACZIUS, LASZLO, DE

(73) Propriétaire/Owner:
SAP AKTIENGESSELLSCHAFT, DE

(74) Agent: SMART & BIGGAR

(54) Titre : INTEGRATION D'APPLICATIONS HETEROGENES
(54) Title: INTEGRATION OF HETEROGENEOUS APPLICATIONS



(57) Abrégé/Abstract:

Methods and apparatus for integrating heterogeneous applications. A method includes distributing an event from a first application of a first system to a second application of a second system. The event specifies at least one of an object and an operation in the



(57) **Abrégé(suite)/Abstract(continued):**

second system. Distributing the event causes the second system to handle the event using the at least one of the object and the operation.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
8 January 2004 (08.01.2004)

PCT

(10) International Publication Number
WO 2004/003741 A3

(51) International Patent Classification⁷: **G06F 9/44**

(21) International Application Number:
PCT/IB2003/003259

(22) International Filing Date: 1 July 2003 (01.07.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/188,352 1 July 2002 (01.07.2002) US

(71) Applicant: **SAP AKTIENGESELLSCHAFT** [DE/DE];
Intellectual Property Department, Neurottstrasse 16, 69190
Walldorf (DE).

(72) Inventors: **HAUSER, Andreas**; Breitenbachstrasse 6a,
69234 Dielheim-Horrenberg (DE). **BIHARI, Pranav**;
Horrenberger Strasse 30, 69168 Wiesloch (DE).

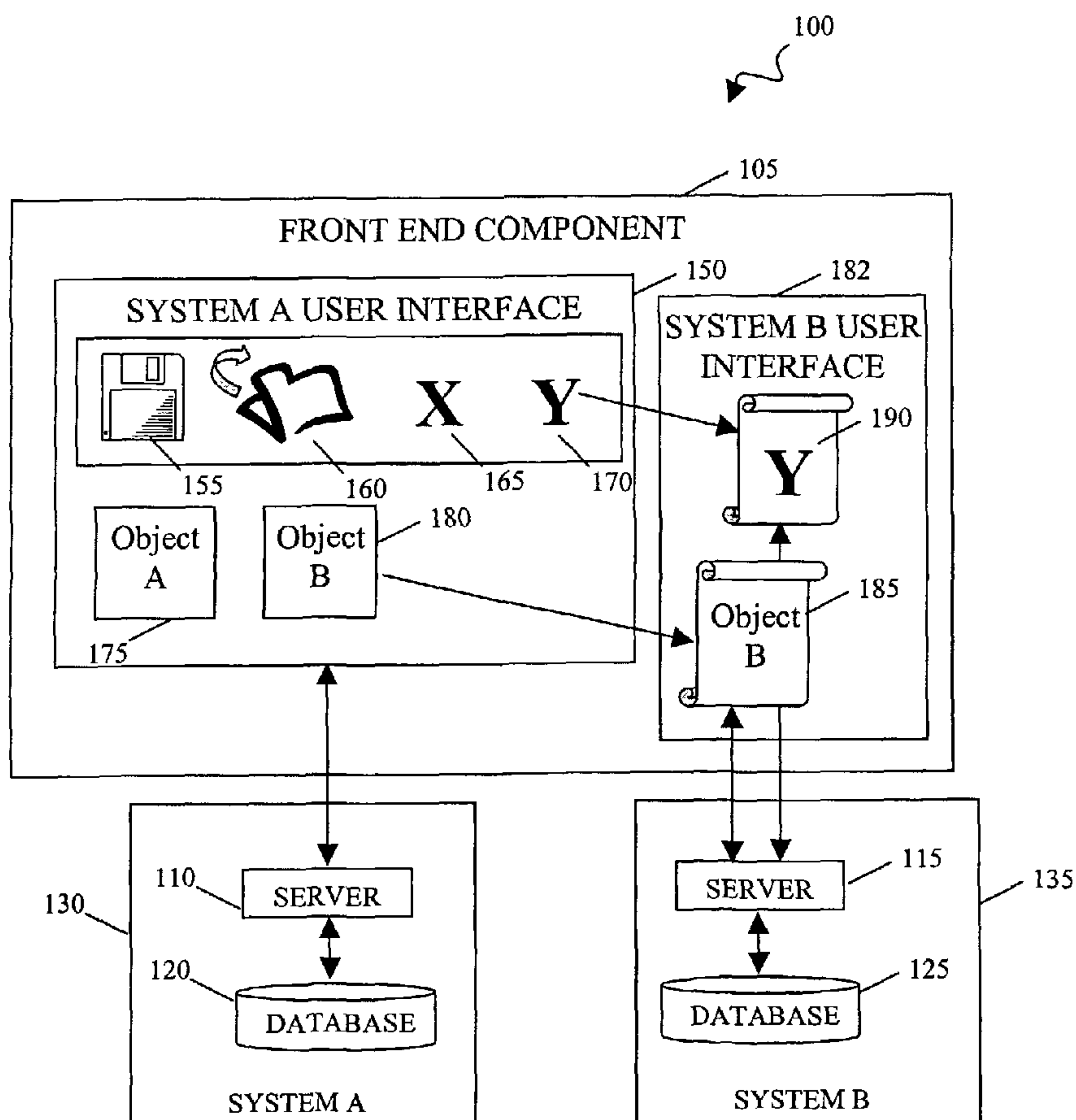
SCHMITT, Bernd; Moltkestrasse 7, 76660 Bruchsal
(DE). **SZERVACZIUS, Laszlo**; Ohrenbergerstrasse 41,
76669 Bad Schoenborn (DE).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,
CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH,
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC,
LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW,
MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC,
SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA,
UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM,
KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO,
SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM,
GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: INTEGRATION OF HETEROGENEOUS APPLICATIONS



(57) Abstract: Methods and apparatus for integrating heterogeneous applications. A method includes distributing an event from a first application of a first system to a second application of a second system. The event specifies at least one of an object and an operation in the second system. Distributing the event causes the second system to handle the event using the at least one of the object and the operation.

WO 2004/003741 A3

WO 2004/003741 A3



Published:

— *with international search report*

(88) Date of publication of the international search report:

2 December 2004

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTEGRATION OF HETEROGENEOUS APPLICATIONS

BACKGROUND

This invention relates to integrating computer programs.

Different types of computer programs can be used to process different business activities in a company. For example, a customer relationship management (CRM) application can be used to process customer relationship activities, a supply chain management (SCM) application can be used to process production activities, and a product lifecycle management (PLM) application can be used to manage information across the whole spectrum of business activities in a company.

A first and second of such applications can have incompatible formats for processes and data. For example, the first application can produce output data in a format that is incompatible with a format of input data for the second application. The first and second applications will be referred to as heterogeneous applications. Furthermore, the first application can reside on a first computer system having formats for processes and data while the second application can reside on a second computer system that has formats for processes and data that is different from the formats of the first computer system. The systems can be produced by different companies that are unwilling or unable to agree to a standard format for processes and data. The first and second computer systems will be referred to as heterogeneous computer systems.

Heterogeneous computer systems can include one or more back-end components, such as a data server, and one or more middleware components, such as an application server. Heterogeneous computer systems can also include one or more front-end components, such as a client computer having a graphical user interface or a Web browser. The components of the system can be interconnected by any form or medium of digital data communication, such as a communication network. Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

Heterogeneous computer systems can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of

computer programs running on the respective computers and having a client-server relationship to each other.

A user will often interact with two or more heterogeneous computer systems from a single front end component that can be part of both heterogeneous computer systems. For example, the user can, through a Web browser on a desktop computer, interface with two heterogeneous applications that are executed by heterogeneous computer systems.

SUMMARY

The present invention provides methods and apparatus, including computer program products, for integrating heterogeneous applications.

In general, in one aspect, there is provided a method for integrating at least a first and a second heterogeneous applications, the first application residing in a first computer system, the second application residing in a second computer system, wherein the first and second heterogeneous applications have different formats for data, and wherein a front end component provides a first user interface enabling accessing data and operations in the first computer system, and a second user interface enabling accessing data and operations in the second computer system within the front end component; the method comprising: distributing an event from the first application of the first computer system to the second application of the second computer system at the front end component, the event specifying at least one data object in the second computer system, the distributing the event from the first application comprising formatting the event using a protocol shared with the second computer system and distributing the formatted event to the second user interface, distributing a key to identify the at least one data object specified by the event, and furthermore, causing the second computer system to handle the event using the at least one data object; wherein the method further comprises accessing a data mapping system that describes the relationship between data in said first and second heterogeneous applications; and mapping the key from a key that is valid in the first computer system to a key that is valid in the second computer system; mapping a field or other data of the at least one data object specified by the event from a field or other data that is valid in the first computer system to corresponding field or other data that is valid in the second computer system; and wherein causing the second computer system to handle the event comprises causing the second user interface to handle the event using the data object identified by the mapped key and using the mapped field or other data.

Implementations of this or any other aspect can include one or more of the following features. Distributing the event can include formatting the event using a protocol shared with the second system by, e.g., formatting the event to a semantic protocol standard shared with the second system. The semantic protocol can be, e.g., one of ebXML (Electronic Business using eXtensible Markup Language), EDI-X12 (Electronic Data Interchange based on American National Standards Institute Accredited Standards Committee X12), and WSDL/UDDI (Web Service Definition Language / Universal Description, Discovery and Integration). Distributing the event can include distributing a key that can be mapped to identify data in the second system. The event can be distributed from a user interface to, e.g., a Web service.

The method for integrating heterogeneous applications can also include receiving user input specifying the event, receiving the event from the first system, parsing the received event, or receiving a key of the object from the second system.

In general, in another aspect, there is provided a computer program product, tangibly stored on machine readable media, for integrating at least a first and a second heterogeneous applications, the first application residing in a first computer system, the second application residing in a second computer system, wherein the first and second heterogeneous applications have different formats for data; the product comprising instructions to cause a processor to: render a first user interface for accessing data and processes in the first computer system at a front end component; render a second user interface for accessing data and processes in the second computer system at the front end component; receive user input through the first user interface, the user input specifying at least one data object from the second computer system; distribute an event to the second user interface to identify the specified data object to the second user interface to cause the second computer system to access the specified data object, distribute a key of the at least one data object specified by the event to the second user interface, the product comprising further instructions to cause the processor to: access a data mapping system that describes the relationship between data in said first and second heterogeneous applications; map the key from a key that is valid in the first computer system to a key that is valid in the second computer system; map a field or other data of the at least one data object specified by the event from a field or other data that is valid in the first computer system to corresponding field or other data that is valid in the second computer system; and cause the second user interface to handle the event using the data object identified by the mapped key and using the field or other mapped data, and wherein the product includes further instructions to cause the

processor to: format a communication using a protocol shared with the second user interface to identify the specified data or process to the second user interface.

5 The instructions can cause the processor to format a communication using a protocol shared with the second user interface to identify the specified data or process to the second user interface or to receive the specified data or process, or a result of performing the specified process on the specified data, from the second system. A technical protocol that is shared with the second user interface can be used to identify the specified data or process to the second user interface, or one of a name, an attribute name, and an object name can be used to identify the specified data or process to the second user interface. Also, one of an XML (eXtensible Markup Language) string and an XML
10 file can be used to identify the specified data or process to the second user interface.

The instructions can cause the processor to use a semantic protocol shared with the second user interface to identify the specified data or process to the second user interface. The semantic protocol can be, e.g., one of ebXML, X12, and UDDI. The instructions can cause the processor to identify the specified data or process to a second graphical user interface.

15 The instructions can cause the processor to render a user interface in a browser page at the front end and identify the specified data or process to a second user interface in the same browser page. The first user interface can be rendered at a front end client.

The invention can be implemented to realize one or more of the following advantages. A system in accordance with the invention integrates heterogeneous applications. The system
20 facilitates interactions of the heterogeneous applications at a front end component of a computer system. The system can achieve integration without tight backend integration of the applications, or in the presence of backend integration. In particular, the system defines business objects and/or processes of heterogeneous applications so that the heterogeneous applications can identify and use the business objects and/or processes for further processing.

25 The details of one or more implementations of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is a pictorial representation of integrated heterogeneous applications.

FIG. 2 shows an implementation of a system landscape where heterogeneous applications are integrated.

5 FIG. 3 shows a method in accordance with the invention for integrating heterogeneous applications.

FIG. 4 shows another method in accordance with the invention for integrating heterogeneous applications.

10 FIG. 5 shows an implementation of another system landscape where heterogeneous applications are integrated.

FIG. 6 shows an implementation of another system landscape where heterogeneous applications are integrated.

FIG. 7 shows an implementation of another system landscape where heterogeneous applications are integrated.

15 FIG. 8 shows another method in accordance with the invention for integrating heterogeneous applications.

FIG. 9 shows an implementation of another system landscape where heterogeneous applications are integrated.

20 FIG. 10 shows another method in accordance with the invention for integrating heterogeneous applications.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

As shown in FIG. 1, a system landscape 100 includes a front end component 105, application servers 110, 115, and databases 120, 125. Front-end component 105 can be a
25 client computer and can provide one or more interfaces for a user to access data and operations of one or more systems. Front-end component 105 is typically a computer, a terminal, or a personal digital assistant (PDA) that communicates with application servers 110, 115 over a communications network. Application server 110 of a system A 130 executes computer programs that process a first activity. Application server 115 of a
30 system B 135 executes computer applications that processes a second activity. The first and second activities can be, for example, business related activities such as project development and product lifecycle management. Database 120 stores data that is used by system A 130 to process the first business activity and database 125 stores data that is

used by system B 135 to process the second business activity. The computer programs that the application servers execute can include business applications.

Front end component 105 receives instructions from system A 130 that relate to the creation and display of a system A user interface 150 and, furthermore, allow access to operations and data in system A 130 from front end component 105. For example, user interface 150 includes selectable command icons 155, 160, 165, 170 that allow a user to specify operations that are to be performed on data. Command icon 155 allows a user to specify that data is to be saved, command icon 160 allows a user to specify that data is to be opened or displayed, and command icons 165, 170 allow a user to specify other operations that are to be performed. User interface 150 also includes data icons 175, 180. Data icon 175 identifies a data object A and data icon 180 identifies a data object B. User interface 150 allows a user to select the data object upon which operations are to be performed using data icons 175, 180.

Front end component 105 receives instructions from system B 135 that relate to the creation of a system B user interface 182. System B user interface 182 allows access to operations and data in system B 135 from front end component 105. For example, system B user interface 182 allows a user to access an operation 185 from system B 135 and data object B 185 from system B. Operation 185 is identified by command icon 170 on user interface 150. Data object B 185 is identified by data icon 180 on user interface 150.

System A user interface 150 and system B user interface 182 can be any of a number of different interfaces or user interfaces that execute on front end system 105. For example, interfaces 150 and 182 can be user interfaces that execute on an operating system, portals or similar display environments, Web service applications that interact with each other, or interfaces built with Web service user interfaces with .NET iViewsTM or Java iViewsTM.

System B 135 is loosely integrated with system A 130 at front end component 105. In particular, system A user interface 150 distributes information, such as events, to system B user interface 182 at front end component 105 to access operations and data from system B 135. An event includes information that specifies one or more data objects and one or more operations associated with the one or more data objects. Interfaces 150, 182 can communicate events in one or both directions using standardized communication formats. This exchange of information between user interfaces 150, 182 at front end component 105 allows a user who interacts with system A user interface 150 to call

WO 2004/003741

PCT/IB2003/003259

operations and retrieve data from system B 135 without requiring tight standardization in the formats of objects and instructions in middleware and/or back end components. For example, system B 135 can be integrated with system A 130 even though they are produced by different manufacturers or by the same
 5 manufacturer at different times when formats have changed. Alternatively, the information exchanged can be other types of information such as index keys and metadata.

FIG. 2 shows one example of front end component 105 in which system A 130 and system B 135 are integrated by exchanging events at a front end component such as front end component 105. Front end component 105 includes a display panel 200 that can be, for example, a browser window, a portal page, or
 10 other area for the display of user interfaces that originate from heterogeneous applications. Display panel 200 includes discrete subsections 205, 210 that, along with system A user interface 150 and system B user interface 182, display a self-contained selection of content to a user. For example, subsection 205 displays a table to a user and subsection 210 displays a graph to a user. Subsections 205, 210, system A user interface 150 and system B user interface 182 can be, e.g., HTML frames, portlets, or iViews displayed
 15 within display panel 200.

In operation, system A 130 and system B 135 can be integrated by exchanging events between system A user interface 150 and system B user interface 182. In particular, system A user interface 150 can act as an event source, while system B user interface 182 can act as an event sink. The exchange of events, and hence the coupling between system A 130 and system B 135, can occur at different eventing levels.
 20 For example, system A user interface 150 can distribute an event to which system B user interface 182 subscribes, so that the event is handled by system B user interface 182. Alternatively, a Web service or a other intermediate can be used to exchange events.

FIG. 3 shows a method 300 whereby system A user interface 150 and system B user interface 182 exchange an event to trigger an action. System A user interface 150 receives user input that selects an event
 25 (step 305). For example, the event can specify that data from data object 185 is to be displayed in system B user interface 182.

System A user interface 150 formats the event for communication with system B user interface 182 (step 310). For example, system A user interface 150 can use shared technical and semantic protocols to communicate the event to system B user interface 182. Example technical protocol formats include event
 30 namespace, attribute-value pairs, extensible markup language (XML) string/file, scripting objects such as JavaScript™ or VBScript™, and serialized Java™ objects. Example semantic protocol standards include

ebXML, EDI-X12, and the WSDL/UDDI/Web services framework. Regardless of the specific protocol used, system A user interface 150 converts the event into a form that can be recognized by system B user interface 182 at front end component 105.

System A user interface 150 distributes the formatted event to system B user interface 182 within front end component 105 (step 315). System B user interface 182 receives the formatted event (step 320) and checks to see if it subscribes to the received event (step 325). If system B user interface 182 subscribes to the event, then system B user interface 182 handles the event (step 330). For example, if the event specifies that data from data object 185 is to be displayed in system B user interface 182, then system B user interface 182 retrieves data object 185 and instructions for displaying data object 185 from application server 115 and database 125.

FIG. 4 shows a method 400 whereby system A user interface 150 and system B user interface 182 exchange an object key of an object specified by an event. The exchanged object key need not be valid in both system A 130 and system B 135. For example, the exchanged key can be valid for identifying a record in system A 130, but may be meaningless in system B 135. System A user interface 150 receives user input that selects the event (step 405). For example, the event can specify the display of a data object that is located in a table in system B 135. System A user interface 150 formats the event (step 410) and distributes the formatted event to system B user interface 182 (step 415).

System B user interface 182 receives the formatted event (step 420) and checks to see if it subscribes to the received event (step 425). If system B user interface 182 subscribes to the event, then system B user interface 182 identifies the system A key to the object specified by the event (step 430). The identified key can be valid only in system A 130, or the identified key can be valid in both system A 130 and system B 135. If necessary, system B user interface 182 maps the key into a second key that is valid in system B 135 (step 435). In addition to mapping a key, system B user interface 182 can map a field or other data that is valid in system A 130 to a field or other data that is valid in system B 135. In order to map a key or other data, system B user interface 182 can access a data mapping system that describes the correspondence between data in system A 130 and system B 135, as discussed further below. System B user interface 182 handles the event using the data object identified by the mapped data (step 440).

Alternatively, system B user interface 182 can format the mapped key or other data and return the formatted, mapped key or data to system A user interface 150. System B user interface 182 can use the same technical and semantic protocols that were used by system A 130 to format the event. System A user interface 150 can then employ the mapped key or data to directly access the object in system B 135, as needed.

FIG. 5 shows system landscape 100 where data mapping can be performed to integrate system A 130 and system B 135 at front end component 105. System A database 120 can include a table 505 with a primary key field 510 and a data record 515. System B database 125 can include a data table 520 with a primary key field 525 and a data record 530. Data table 505 can describe, e.g., the technical details of products for a supply chain management (SCM) system while data table 520 can describe sales information about products for a customer relationship management (CRM) system.

The format of the contents of the data records in both tables 505, 520 need not be identical in primary key fields 510 and 525. For example, data record 515 can contain a product number (e.g., "Widget 883") in primary key field 510, while data record 530 can contain a brand name (e.g., "Value Widget") in primary key field 525. This discrepancy can prevent system A user interface 150 from directly accessing the content of data table 520.

In this case, system landscape 100 can also include a data mapping system 535 that includes information about the relationship between data in heterogeneous applications. Data mapping system 535 can include information for mapping keys or mapping fields, and can be implemented, e.g., as a database of relationships or a collection of rules that describe the relationships between data. For example, data mapping system 535 can describe the relationships between the data in key field 510 and key field 525 in, e.g., a table that identifies the correspondence between the product numbers in key field 510 and the brand names of the products in key field 525. System B user interface 182 can access data mapping system 535 to map the key received from system A user interface 150 and handle the event.

FIG. 6 shows an alternative system landscape 600 where data mapping can be performed to integrate system A 130 and system B 135 at front end component 105. System B database 125 can include a data table 605 with a primary key field 610 and data records 615, 620, 625. Data table 605 can describe the same products described in data table 505 but using different information. For example, data table 605 can describe the technical details of various components used to assemble the products described in data

table 505. The description can include detailed information about the components that is not included in data table 505.

The data in data table 605 is keyed to the component name in primary key field 610 and not to the associated product number (or associated product numbers) in primary key field 510. Therefore, when an event includes the key from primary key field 510 in system A 130, system B 135 maps the key to the component name in primary key field 610. For example, if a user requests the display of the weight of the widget head of widget in record 415, the product number from key field 410 is mapped to identify which widget head record 615, 620, 625 is associated with the widget in record 415.

Furthermore, system B user interface 182 need not perform the data mapping in order for system A 130 and system B 135 to be integrated at front end component 105. For example, data mapping can be performed by system B application server 115. In particular, system B application server 115 can receive an unrecognized key from system B user interface 182 and then map the unrecognized key using a data mapping system 630. System B application server 115 can then return an object identified by the mapped key to system B user interface 182.

FIG. 7 shows a system landscape 700 that includes a Web service intermediate to exchange events between system A user interface 150 and system B user interface 182. In addition to front end component 105, application servers 110, 115, and databases 120, 125, system landscape 700 also includes a portal server 705 and a portal storage 710. Portal server 705 executes instructions on portal storage 710 to invoke a Web service 715. Web service 715 is an application component that is accessible by system A user interface 150 and system B user interface 182 using shared protocols such as, e.g., UDDI, SOAP, XML, HTTP, or other Internet protocols. Web service 715 can be, e.g., a self-contained, modular application that can be published, located, and invoked across a communications network such as the Internet. Web service 715 can perform functions such as, e.g., publishing an event and subscribing to an event.

In operation, Web service 715 can be invoked by system A user interface 150 and system B user interface 182 to exchange events. As shown in FIG. 8, in particular, system A user interface 150 can receive user input that selects the event (step 805). System A user interface 150 can publish the event to Web service 715 using a shared protocol (step 810).

Web service 715 receives the event from system A user interface 150 (step 815) and in turn publishes the event to system B user interface 182 (step 820). Web service

715 can publish the event using the same protocol, or using another protocol that is shared with system B user interface 182.

System B user interface 182 receives the event from Web service 715 (step 825) and checks to see if it is subscribed to the event (step 830). If subscribed, system B user interface 182 handles the event (step 835).

In one implementation, front end component 105 receives instructions from portal server 705 that relate to the creation and display of system A user interface 150 and system B user interface 182. In this case, portal server 705 receives user input over system A user interface 150 and distributes an event to system B user interface 182, which in turn accesses system B 135 to handle the event.

FIG. 9 shows a system landscape 900 that includes display panel intermediate 200 to exchange events between system A user interface 150 and system B user interface 182. In addition to front end component 105, application servers 110, 115, and databases 120, 125, system landscape 900 also includes a panel system 905 including a panel server 910 and a panel storage device 915. Panel server 905 executes instructions on panel storage 910 to create display panel 200 at front end component 105. As described above with regards to FIG. 2, display panel 200 can include system A user interface 150 and system B user interface 182 as, e.g., portlets, iViews, or HTML frames. System landscape 900 also includes a data mapping system 920 that is accessible by panel server 905.

In operation, system A user interface 150 and system B user interface 182 can exchange events using display panel 200 as an intermediate to achieve front end integration. As shown in FIG. 10, in particular, system A user interface 150 can receive user input that selects the event (step 1005). System A user interface 150 can publish the event to panel 200 using a shared protocol (step 1010).

Display panel 200 receives the event from system A user interface 150 (step 1015). If necessary, display panel 200 accesses data mapping system 920 to map the data from system A 130 to data that is valid in system B 135 (step 1025). For example, display panel 200 can map a key or a field from system A 130 to a key or a field that is valid in system B 135. Instead of accessing data mapping system 920 to map the data from system A 130 to data that is valid in system B 135, display panel 200 can access any location where mapping information, including mapping rules, can be found. Display panel 200 also publishes the event, with the original data or with a mapped data, to system B user interface 182 (step 1030).

System B user interface 182 receives the event from display panel 200 (step 1035) and checks to see if it is subscribed to the event (step 1040). If subscribed, system B user interface 182 handles the event (step 1045).

The following selected scenarios provide example illustrations of the methods and apparatus of implementations of the invention.

SCENARIO 1:

Two user interfaces pass data from two systems at a front end client. The two user interfaces have a defined, shared syntax for defining objects and a defined, shared technology to exchange the objects at the front end client.

The objects can be defined using, e.g., an XML definition, and the objects can be communicated by passing, e.g., an XML string between the interfaces. Alternatively, serialized Java™ objects can be exchanged between the interfaces.

The objects can be exchanged based on standards like EDI-X12 and ebXML. Alternatively, a Java™ serialized object, an XML string, or a JavaScript™ object can be modeled as an X12 file or as an ebXML message.

SCENARIO 2:

Two user interfaces pass data at a front end client using a portal solution such as the Enterprise Portals Client Framework (EPCF). In EPCF eventing, a serialized Java™ object can have a structure that is similar to the structure of an X12 file.

A single browser page can display a first user interface from a first vendor in a first frame a second user interface from a second vendor in a second frame. The user interfaces can also be, e.g. iView's in a portal. When the first user interface is rendered, a link is included. The link includes a serialized data object for eventing in, e.g., an X12 structure.

When a user clicks on this link in the first user interface, the second user interface receives and handles the event of the serialized data object. An application server for the second user interface converts the serialized data object into X12 standard.

SCENARIO 3:

Two interfaces communicate at a front end component without passing data between the systems. The two interfaces have a standardized language/protocol to communicate at the front end component. An interaction with one interface triggers an event to which the other interface reacts.

For example, a first interface receives user input requesting that a new favorite be added to a list of favorites. The first interface searches for the new favorite and calls a

function indicating that the new favorite is to be added to the list of favorites. This function call is communicated to the second interface at the front end component, which subscribes by updating the list of favorites.

SCENARIO 4:

5 Two interfaces manage object information for two heterogeneous systems. Object identification can be transferred between the interfaces as a parameter of the event.

For example, a first system manages operational project data, while a second system stores additional costing and resource information. When a user selects a project using a first interface of the first system, the first interface triggers an event that transfers
10 the project number to a second interface of the second system, in the front end.

The second interface receives the event, reads the project number, and accesses the second system to display the additional cost or resource data. If both systems use the same project number as a key, no key mapping is required. On the other hand, if both systems use a different key to identify the project, the second interface (or other
15 component of the second system) can map the key to identify and display the additional cost or resource data.

SCENARIO 5:

Only a second system stores a particular, complex data object. A first interface of a first system can retrieve the data object from the second system so that it may be
20 transferred into the first system. The data object may be formatted and communicated as a serialized structure using, e.g., XML.

For example, a second system can store a bill of material that includes multiple sub-assemblies. A particular sub-assembly from the bill of material is to be transferred into the first system. When a user selects the sub-assembly using the first interface, the
25 first interface triggers an event that is distributed to the second interface. The second system handles the event, and transfers the sub-assembly to the first interface as a serialized string in an XML format, such as, e.g., ebXML. The first interface receives the serialized string, parses the XML format and creates a bill of material or a document structure for the sub-assembly within the first system.

30 The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a

programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

Method steps of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus of the invention can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

To provide for interaction with a user, the invention can be implemented on a computer having a display device such as a CRT (cathode ray tube) or LCD (liquid crystal display) monitor for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback,

such as visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

The invention can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the invention, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

A number of implementations of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, the steps of the invention can be performed in a different order and still achieve desirable results.

Two or more applications can be integrated by exchanging information at a user-interface level. Integration can be implemented with exchanges of any types of information, including events, index keys, metadata, or any combination thereof. An application generally can send and receive information to and from other applications through user interfaces, such as user interfaces 150 and 182. The exchange of information between or among heterogeneous applications can be implemented through, for example, JavaScript commands. Furthermore, the user interface can be established at any component level in a computer system. For example, user interfaces can be created and displayed by a middleware component of a computer system.

A user interface such as user interface 150 can include a variety of user input mechanisms through which a user can select operations or data in a heterogeneous application. For example, rather than or in addition to icons, a user interface can include menus such as pull down menus, or link identifiers such as buttons through which a user can select operations or data in a heterogeneous application. Interfaces for interacting

with the two or more heterogeneous applications can be provided by, for example, an operating system, a browser, or a portal.

5 The exchange of information between or among heterogeneous applications can occur at a client side. The exchange of information, in these cases, does not require a integration on a server side.

10 The information that is exchanged can include information other than events. When events are exchanged, the events can specify only data, only operations, or both objects and operations. Events specify only data when there is no need to specify an operation. For example, system B user interface 182 can be dedicated to performing a single operation. System A user interface 150 can distribute events or other communications that only identify data (e.g., data objects) upon which system B user interface 182 is to perform the single, dedicated operation.

Information, such as metadata exchanged at the front end component, can be used by the backend systems to interact at the backend using a standard such as, e.g., X12.

15 An event can be broadcast to several user interfaces simultaneously and only handled by those user interfaces that are subscribed to the broadcast event. Exchanges of events can be effected by the heterogeneous applications, by the interfaces of the individual heterogeneous applications, by the general interface (as provided by the browser or portal) that includes the individual interfaces, or any combination thereof. The frames in a portal, for example, can broadcast and receive events.

20 An application can determine which other applications are to receive information, such as events, which the application sends. Alternatively, the application can broadcast information, such as events, to other applications and the other applications determine whether the information should be processed. Alternatively, there can be a computer program product that is dedicated to managing the exchange of information so that the applications with which a user is interfacing does not have to manage such exchange. Accordingly, other implementations are within the scope of the following claims.

What is claimed is:

CLAIMS

1. A method for integrating at least a first and a second heterogeneous applications, said first application residing in a first computer system, said second application residing in a second computer system, wherein said first and second heterogeneous applications have different formats for data, and wherein a front end component provides a first user interface enabling accessing data and operations in the first computer system, and a second user interface enabling accessing data and operations in the second computer system within the front end component, the method comprising:

distributing an event from the first application of the first computer system to the second application of the second computer system at the front end component, the event specifying at least one data object in the second computer system, said distributing the event from the first application comprising formatting the event using a protocol shared with the second computer system and distributing the formatted event to the second user interface,

distributing a key to identify the at least one data object specified by the event, and furthermore,

causing the second computer system to handle the event using the at least one data object;

accessing a data mapping system that describes the relationship between data in said first and second heterogeneous applications; and

mapping the key from a key that is valid in the first computer system to a key that is valid in the second computer system;

mapping a field or other data of the at least one data object specified by the event from a field or other data that is valid in the first computer system to corresponding field or other data that is valid in the second computer system; and

wherein causing the second computer system to handle the event comprises causing the second user interface to handle the event using the data object identified by the mapped key and using the mapped field or other data.

2. The method of claim 1, wherein the event includes information specifying at least one operation in the second computer system, and wherein causing the second computer system to handle the event comprises causing the second user interface to handle the event using further the at least one operation.

3. The method of claim 1 or 2, wherein formatting the event comprises:

formatting the event to a semantic protocol standard shared with the second computer system.

4. The method of claim 3, wherein formatting the event to a semantic protocol standard comprises:

formatting the event to one of an ebXML (Electronic Business using eXtensible Markup Language), an EDI-X12 (Electronic Data Interchange based on American National Standards Institute Accredited Standards Committee X12), and a WSDL/UDDI (Web Service Definition Language / Universal Description, Discovery and Integration) protocol.

5. The method of claim 1 or 2, wherein distributing the event to the second application of the second computer system comprises:

distributing the event to a Web service.

6. The method of claim 1 or 2, further comprising:
receiving user input specifying the event.

7. The method of claim 1 or 2, further comprising: receiving the event from the first computer system.

8. The method of claim 7, further comprising:
parsing the received event.

9. The method of claim 1 or 2, further comprising:

receiving a key of the object from the second computer system.

10. A computer program product, tangibly stored on machine readable media, for integrating at least a first and a second heterogeneous applications, said first application residing in a first computer system, said second application residing in a second computer system, wherein said first and second heterogeneous applications have different formats for data; the product comprising instructions to cause a processor to:

render a first user interface for accessing data and processes in the first computer system at a front end component;

render a second user interface for accessing data and processes in the second computer system at the front end component;

receive user input through the first user interface, the user input specifying at least one data object from the second computer system;

5 distribute an event to the second user interface to identify the specified data object to the second user interface to cause the second computer system to access the specified data object,

 distribute a key of the at least one data object specified by the event to the second user interface,

10 access a data mapping system that describes the relationship between data in said first and second heterogeneous applications;

 map the key from a key that is valid in the first computer system to a key that is valid in the second computer system;

 map a field or other data of the at least one data object specified by the event from a field or other data that is valid in the first computer system to corresponding field or other data that is valid in the second computer system; and

 cause the second user interface to handle the event using the data object identified by the mapped key and using the field or other mapped data, and wherein the product includes further instructions to cause the processor to:

20 format a communication using a protocol shared with the second user interface to identify the specified data or process to the second user interface.

11. The product of claim 10, wherein the user input specifies further a process from the second computer system, the event further identifies the specified process to the second user interface and the second user interface handles the event using further the specified process.

12. The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

 receive the specified data or process from the second computer system.

13. The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

receive a result of performing the specified process on the specified data from the second computer system.

5 14 The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

use a technical protocol shared with the second user interface to identify the specified data or process to the second user interface.

10 15. The product of claim 14, wherein the product includes instructions to cause the processor to:

use one of a name, an attribute name, and an object name to identify the specified data or process to the second user interface.

16. The product of claim 14, wherein the product includes instructions to cause the processor to:

15 use one of an XML string and an XML file to identify the specified data or process to the second user interface.

17. The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

20 use a semantic protocol shared with the second user interface to identify the specified data or process to the second user interface.

18. The product of claim 17, wherein the product includes instructions to cause the processor to:

use one of ebXML, X12, and UDDI to identify the specified data or process to the second user interface.

25 19. The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

identify the specified data or process to a second graphical user interface.

20. The product of claim 10 or 11, wherein the product includes instructions to cause the processor to:

render a user interface in a browser page at the front end component; and identify the
5 specified data or process to a second user interface in the same browser page.

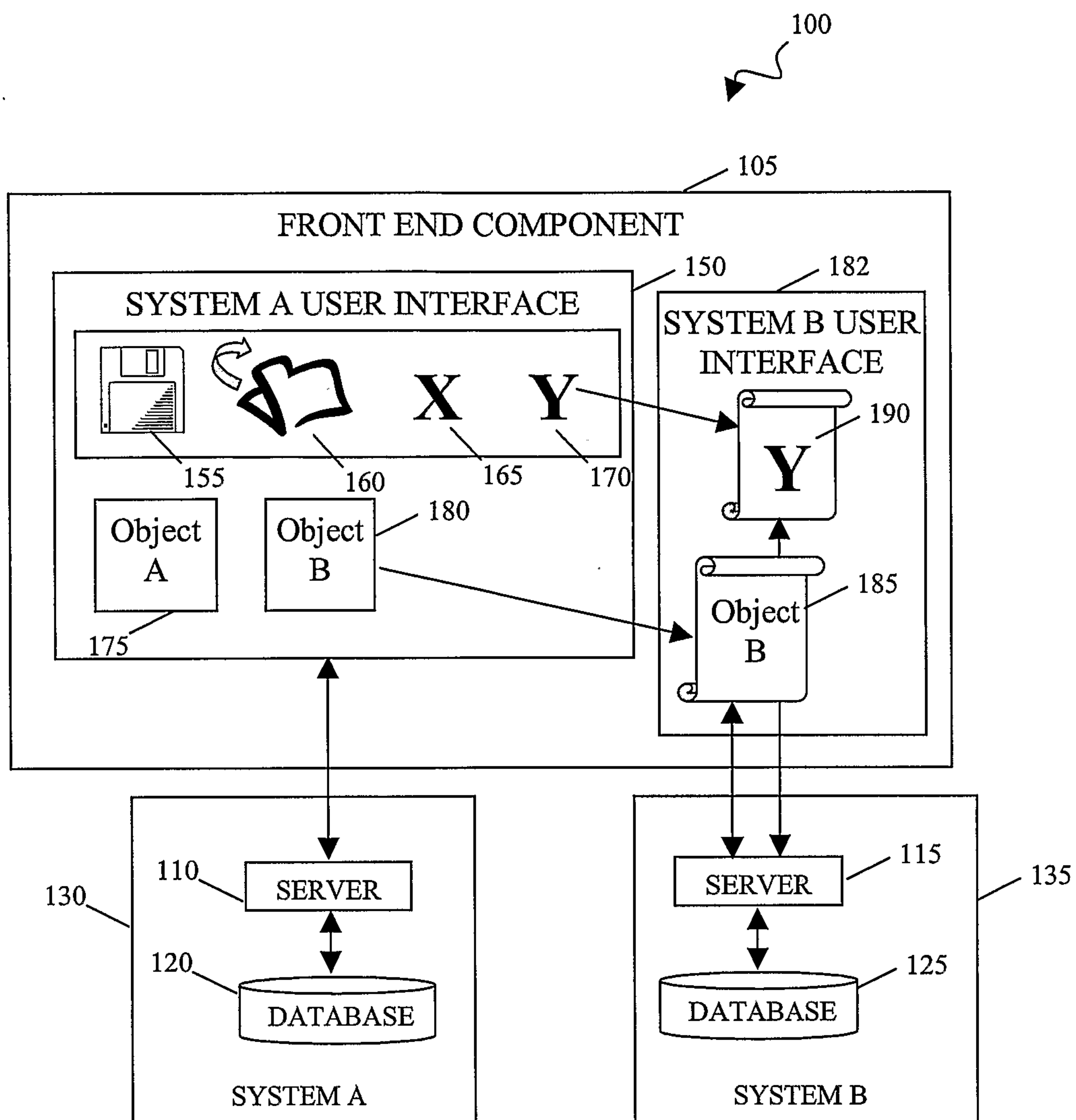


FIG. 1

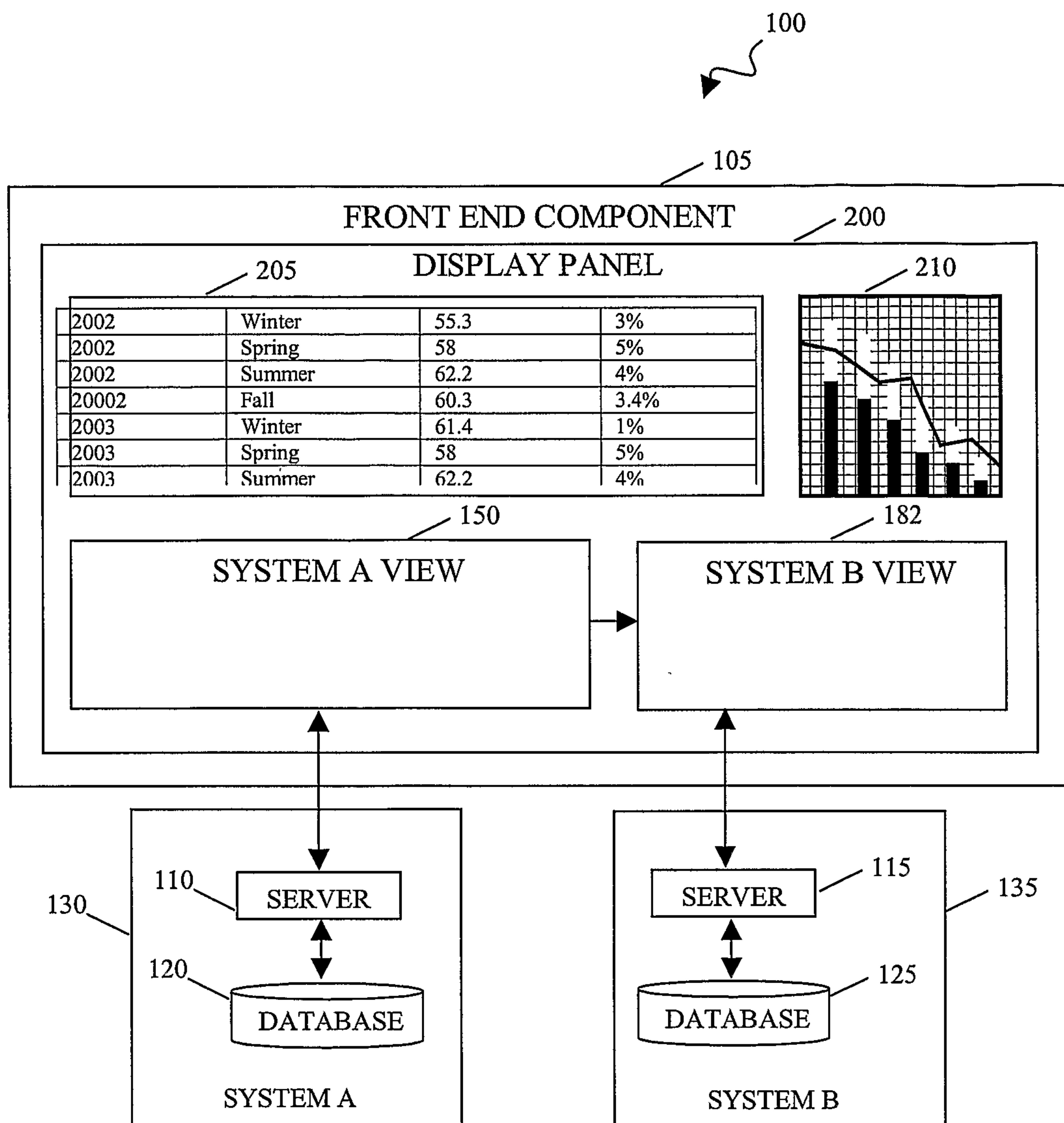
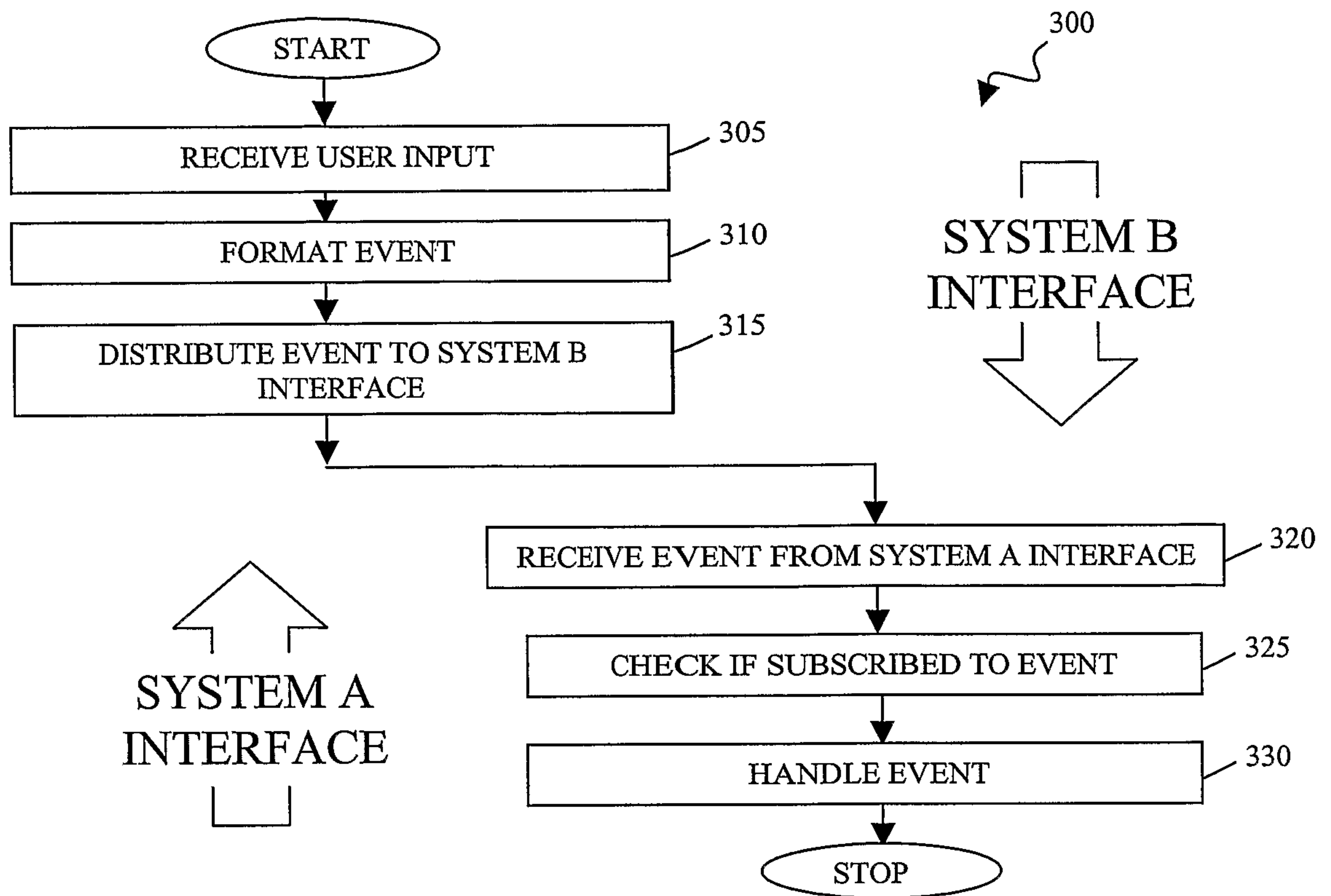


FIG. 2

**FIG. 3**

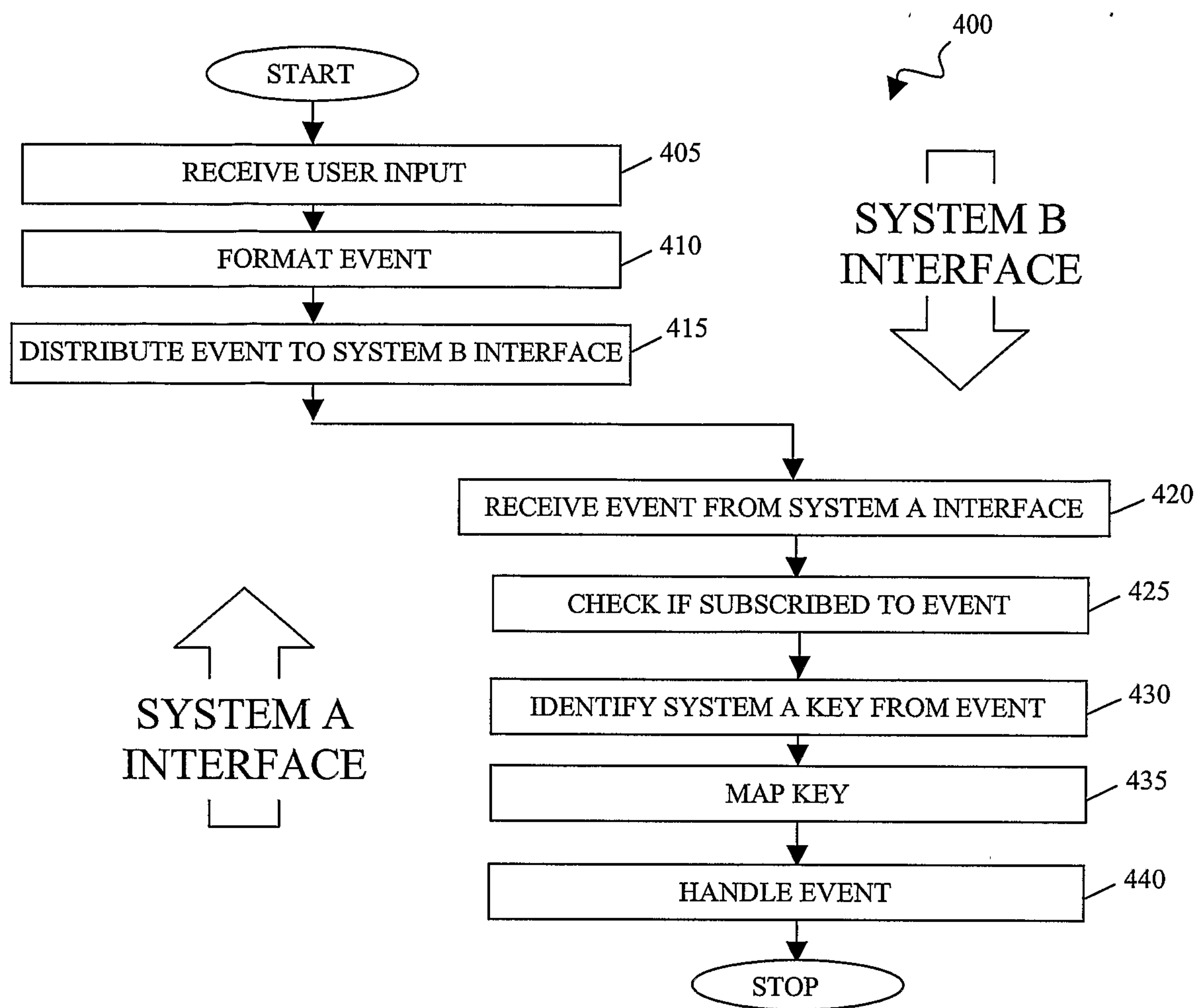


FIG. 4

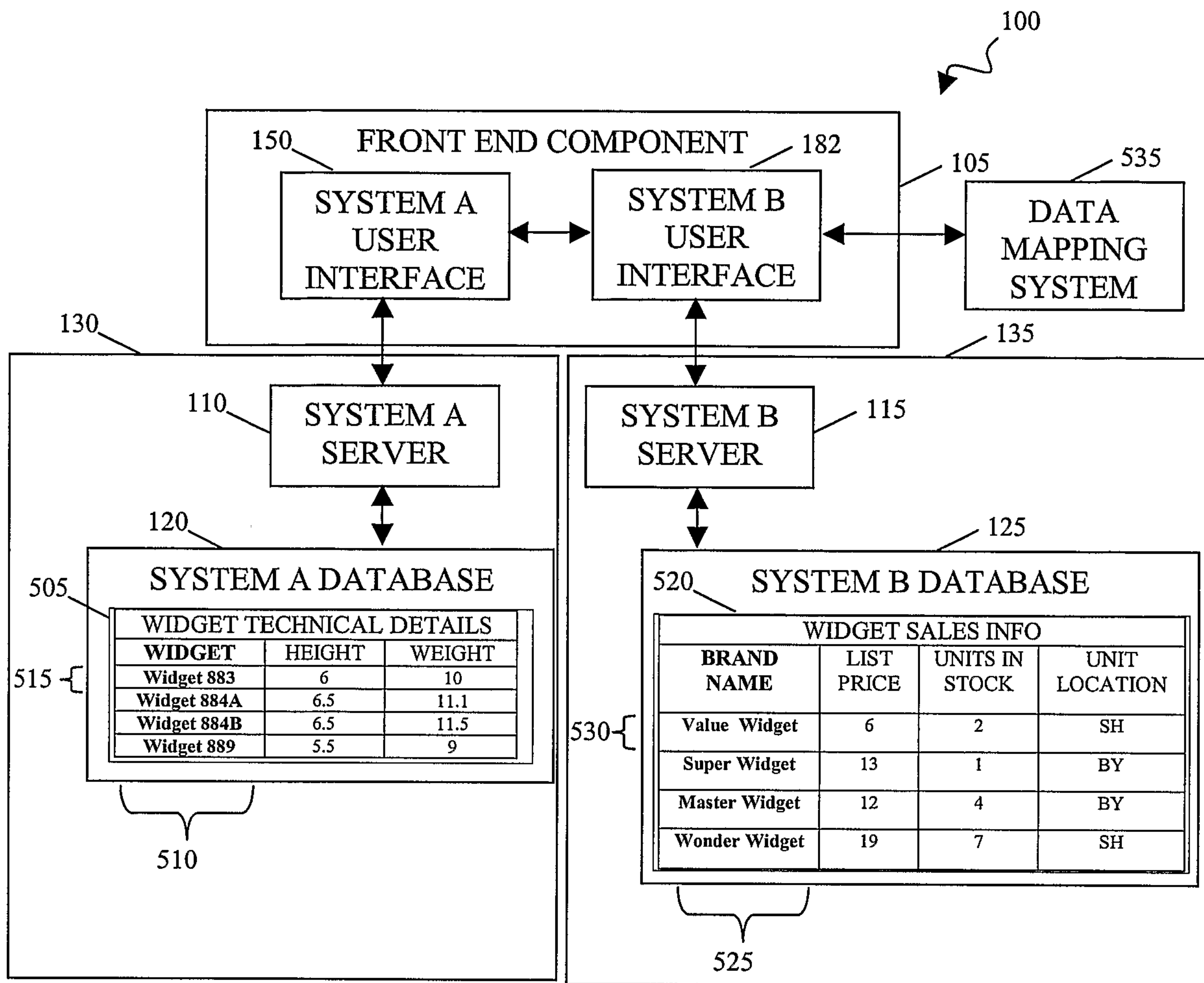


FIG. 5

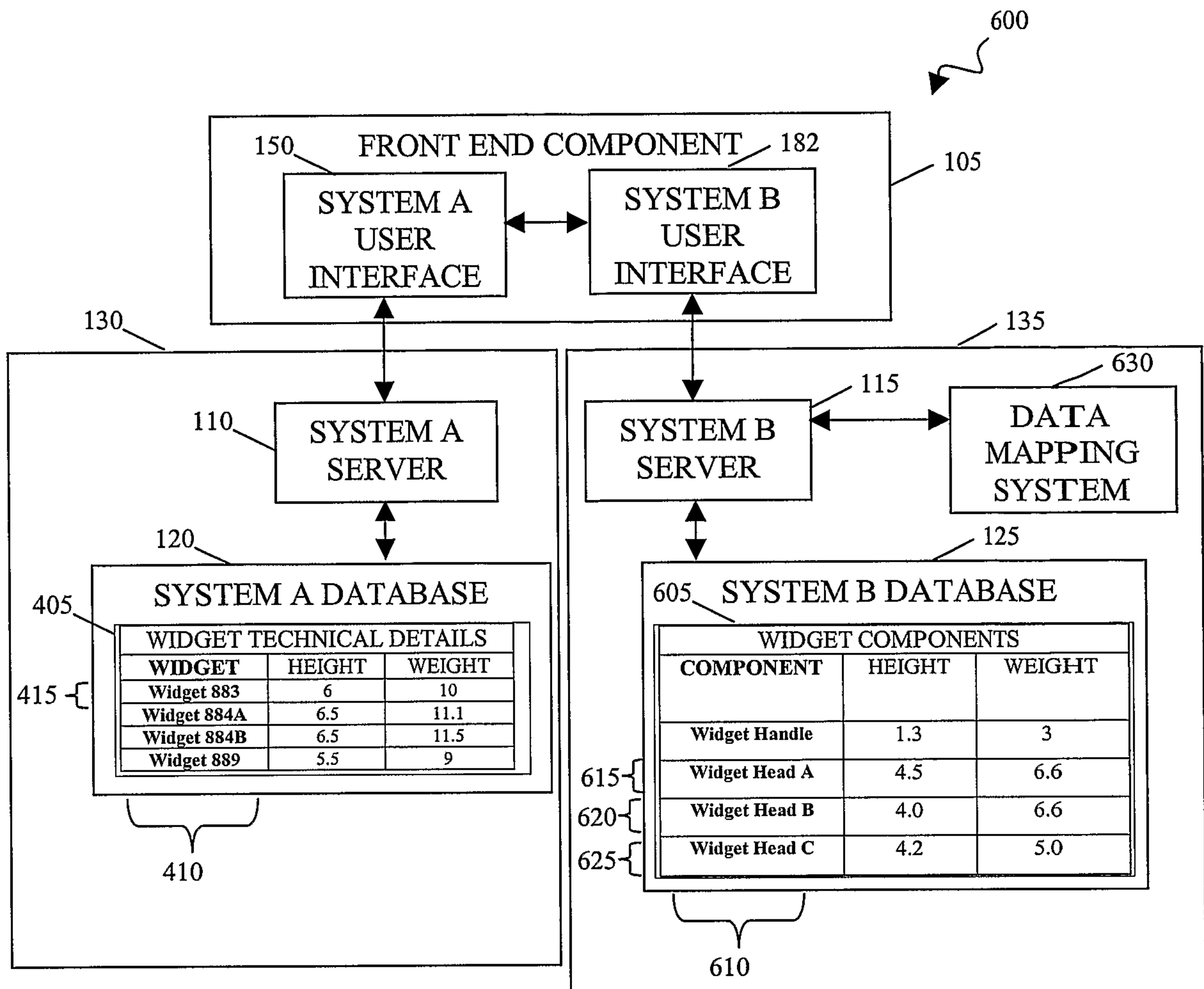


FIG. 6

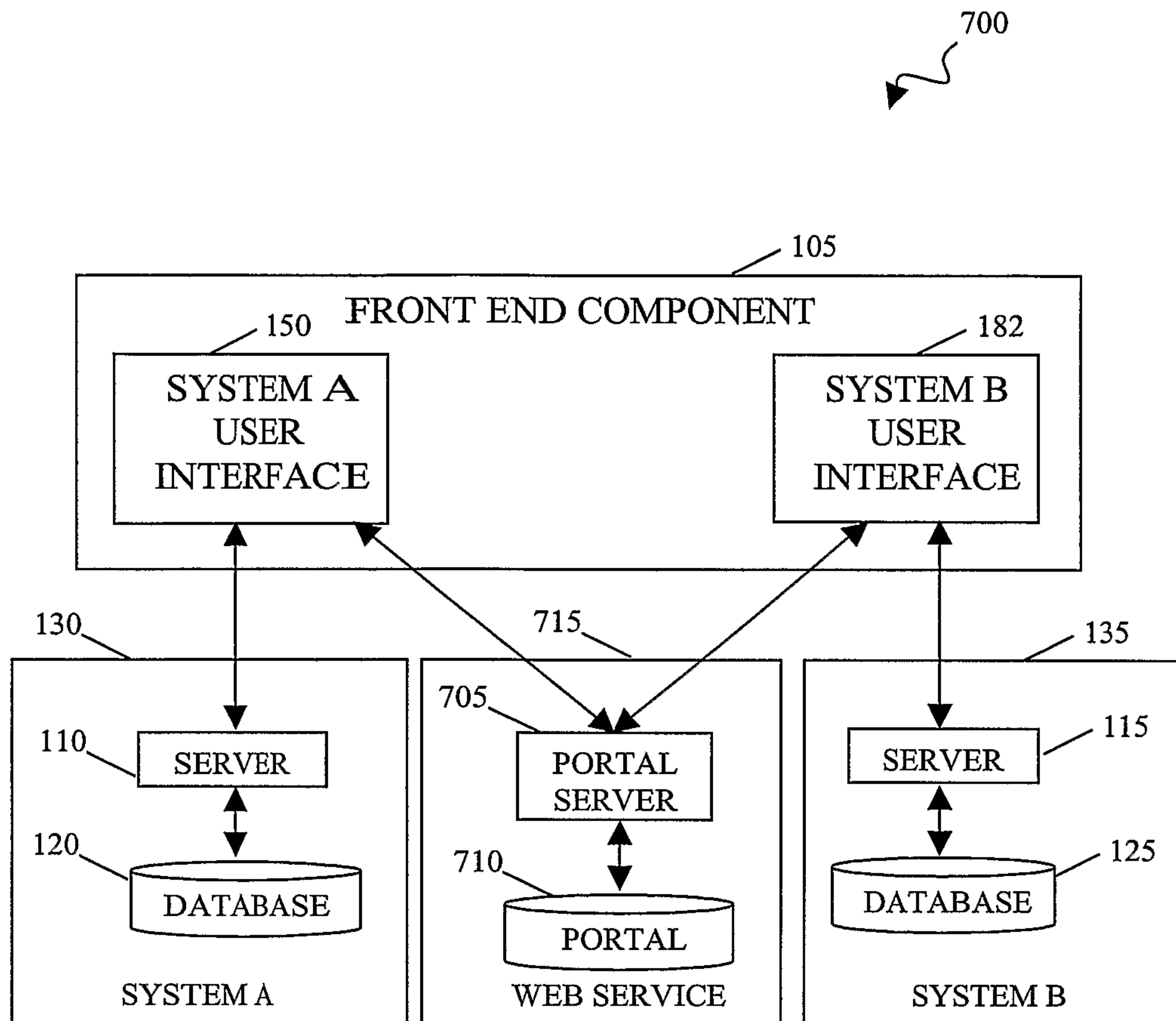


FIG. 7

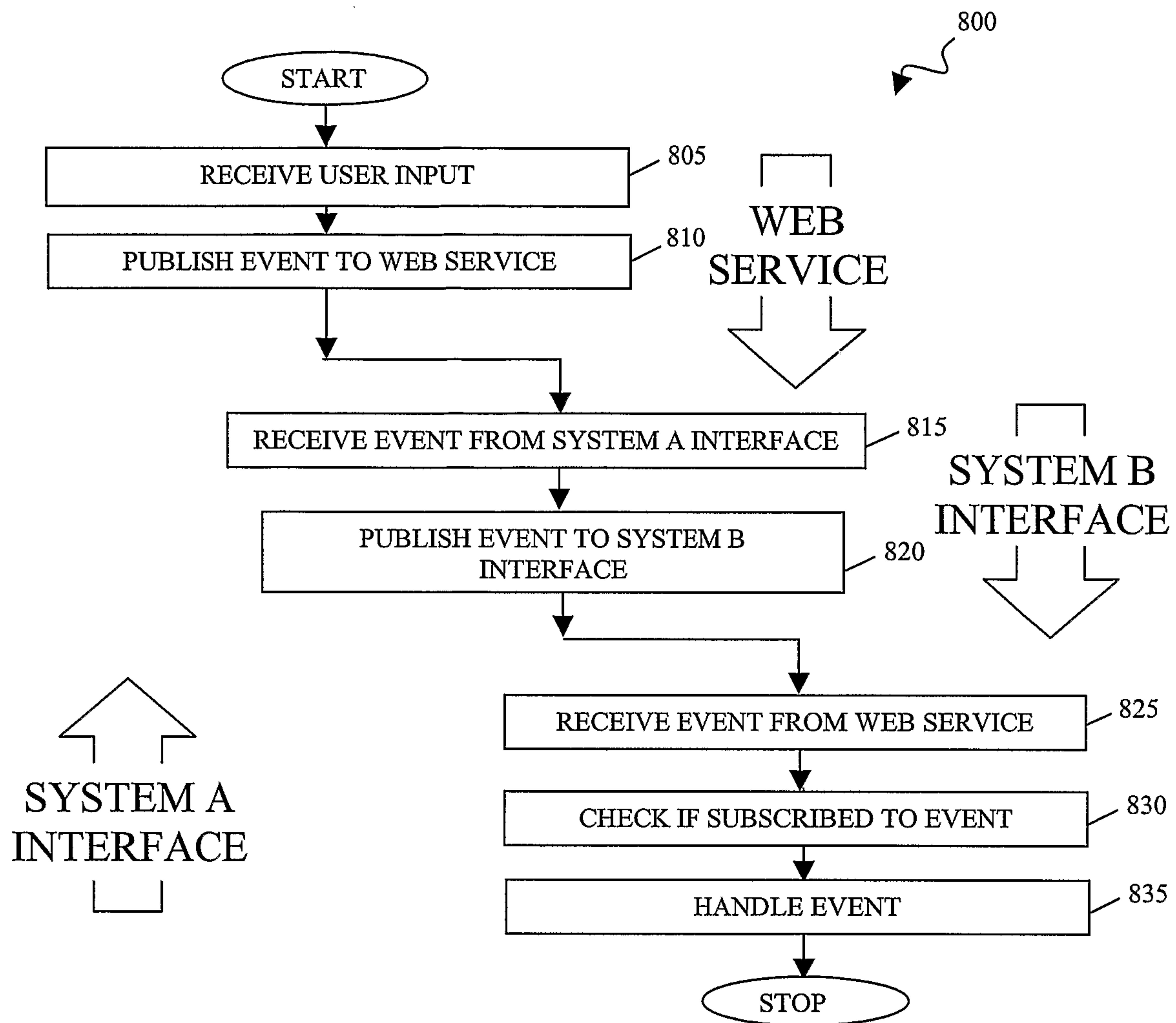


FIG. 8

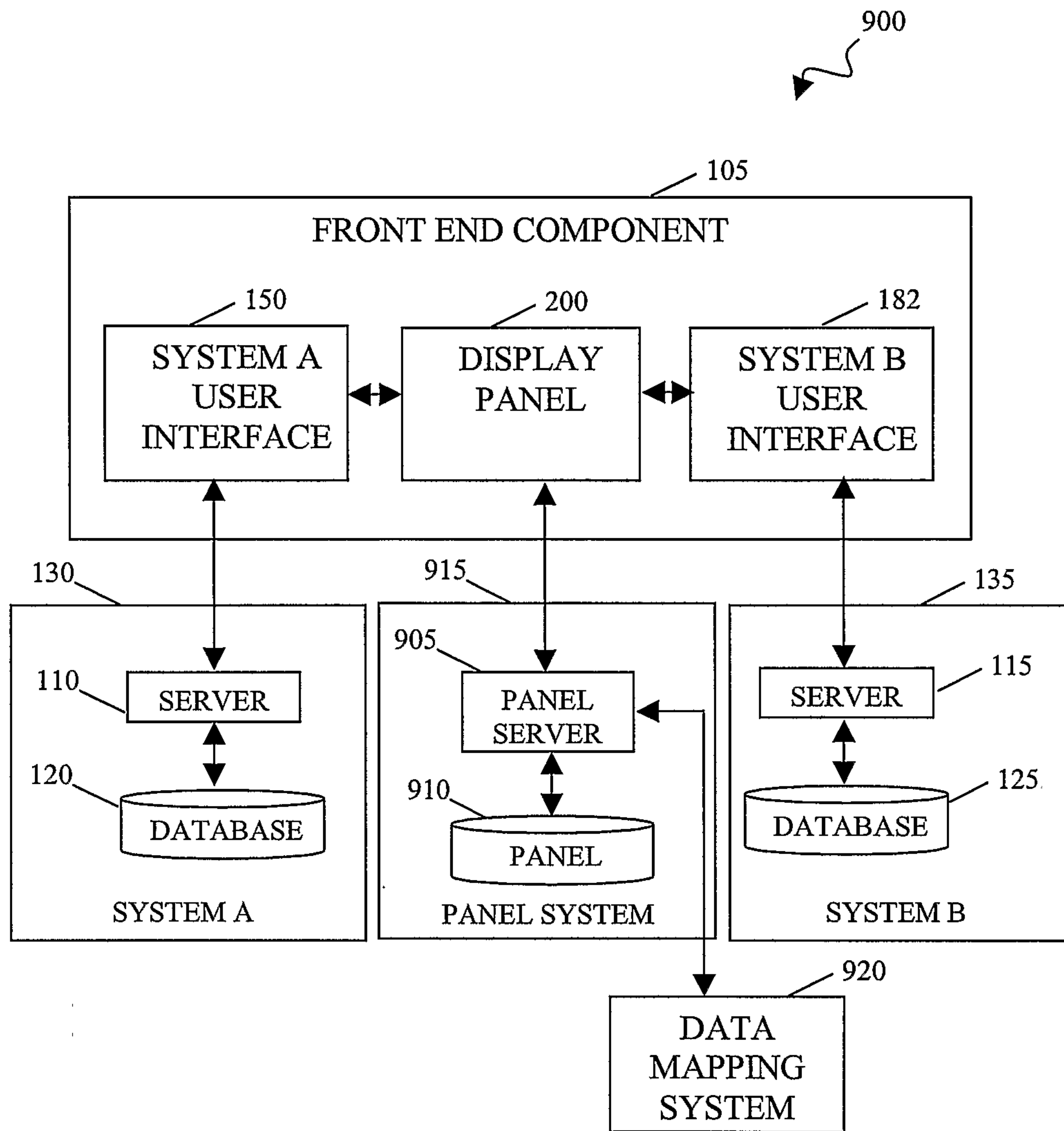


FIG. 9

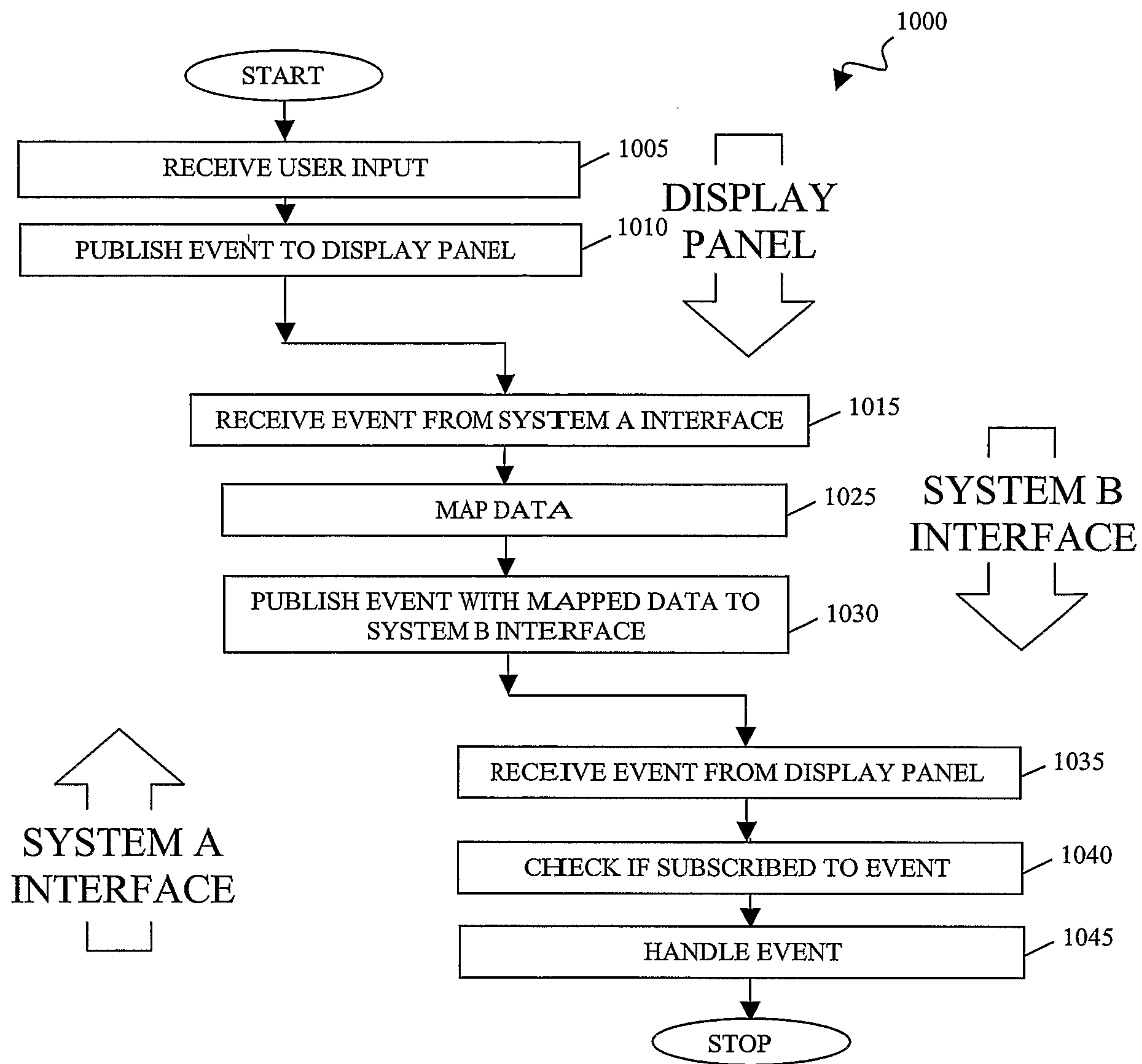


FIG. 10

